ECE 271 – Microcomputer Architecture and Applications Lecture 19

Vince Weaver

http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

4 April 2019

Announcements

- Read Chapter 15.4
- Chuck Peddle on Friday
- Prelim date? Consensus in class was to have it on the 25th
- Servo motor word clock

Lab #9

Input Capture

- Input capture is measuring the time between two transitions on a signal
  - Rising/Rising or Falling/Falling or Rising/Falling or Falling/Rising
  - When transition happens, the current timer count (CNT) is saved to the CCR (Compare and capture register)
  - Also an interrupt (or other event such as DMA) can be triggered

- Time elapsed can be calculated by taking the previous timestamp and subtracting from the current one.

Overflow

- Our timers only have 16 bits. So if programmed at 1us (1MHz) the counter will overflow after 6.5ms.
- What can we do about that?
- In addition to tracking time, count how many times the counter overflows in between.
- How can we do that? Have it trigger an interrupt and count.
- Total time is $overflows * maxtime + (current - last)$

What can we measure?

- Pulse width, period, and duty cycle

Capture Config

- Input can be external pin, or can be another internal timer
- Edge detector: can be only falling, only rising, or both
- Digital Input Filter: (debouncing), number of events that must happen before reporting an event

What it does

- Wait until the number of transitions reaches the threshold in the input capture prescaler bits and the CCMR register, a capture happens
  - The free-running counter CNT is latched into the CCR register for that channel
  - There is a 16-bit capture/compare for each capture channel 1-4
  - In the status register SR the CCIF (capture/compare interrupt flag) for the channel is set

- If the flag was already set, it means we captured again before anyone took care of the last one. In that case the CCOF flag (capture/compare over-capture flag) is set.
- If interrupt is enabled in the CCIE bit (capture/compare interrupt enable bit) then an interrupt is generated
- Other stuff happens if you are doing DMA

Configuring Input Capture

- For this example using Channel 1 of Timer 4
- Select active input: CC1S bits in CCMR1
  - 00: Channel 1 is output (this is what we used for PWM)
  - 01: Input, mapped to timer input 1 (TI1)
  - 10: Input, mapped to timer input 2 (TI2)
  - 11: Input, mapped to TRC (trigger output of other counter)
- Set the input filter – 4 bits. 0 means none

- Set the active edge. CC1P and CC1NP. (double check manual)
  - CC1NP=0 CC1P=0 then rising
  - CC1NP=0 CC1P=1 then falling
  - CC1NP=1 CC1P=1 then both
- Set the input prescaler (can stride, only activating every 1, 2, 4 or 8 events)
- Enable input capture (CC1E bit in CCER)
- Enable the interrupt, TIE trigger interrupt and UIE update interrupt
- Enable the counter, set the CEN enable bit in the CR1

register.

Simple Interrupt Handler

```
volatile uint32_t pulse_width=0,last_captured=0,signal_polarity=0;
void TIM4_IRQHandler() {
    uint32_t current_captured;

    if ((TIM4->SR & TIM_SR_CC1IF)!=0) { // check if irq flag set
        current_captured=TIM4->CCR1;  // auto-clears the CC1IF irq flag
        signal_polarity=1-signal_polarity;
        if (signal_polarity==0) { // only cal chwn low
            pulse_width=current_captured-last_captured;
        }
        last_captured=current_captured;
    }
    if ((TIM4->SR & TIM_SR_UIF)!=0) { // check if oflo has taken place
        TIM4->SR &= ~TIM_SR_UIF;  // clear UIF flag to prevent re-enter
    }
}
```

Input Capture with Automatic Reset

- In Section 15.4.2 if you are curious

Ultrasonic Distance Sensor

- HC-SR04. Generates an ultrasonic signal at 40kHz. Why can't you hear it?
- Receiver detects wave of reflected sound back.
- $Distance = \frac{RoundTripTime \times SpeedofSound}{2}$
- To trigger, send a pulse of 10us to the trigger pin. This makes it send out 8 cycles of 40KHz sound.
- Echo triggers a 5V pulse on the echo pin. Width is proportional to distance.
- $Distance = \frac{PulseWidth(us)}{58} cm$

- $Distance = \frac{PulseWidth(us)}{148} inch$
- Can measure from 2cm to 400cm with a resolution of 0.3cm (pulse width between 150us and 25ms). No echo will give 38ms.
- Hook up 5V, GND. Can take 3.3V trigger. Some inputs 5V tolerant on STM32L.
- Sends 40kHz pulse. Will you hear it?
- Capture time in us. Divide by 148 to get inches.

Actually coding this up

- Use the 16MHz HSI clock
- Set up GPIO PE.11 to trigger every 0.655s with a 10us pulse to activate
  Use PWM mode
- Get the echo result on PB.6 and feed to the measurement code
- Do we have to worry about overflow? Yes.